NISTIR 5819

X.500 Directory Schema Design Handbook

Carol A. Warnar
Advanced Networking Technologies Division

John Tebbutt
Information Access and User Interfaces Division

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

NIST

NISTIR 5819

X.500 Directory Schema Design Handbook

Carol A. Warnar
Advanced Networking Technologies Division

John Tebbutt
Information Access and User Interfaces Division

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

April 1996

Table of Contents

1. Introduction

This document contains a high level schema description including a description of the schema components, the storage of schema information in the Directory Information Tree (DIT), and the tailoring of the schema components to meet an organization's needs. Pilot projects and other work in the area of schema design are reviewed and summarized.

For more detailed information related to X.500 Directory products, please see the National Institute of Standards and Technology Special Publication 500-228, *Guidelines for the Evaluation of X.500 Directory Products* [1], or the ITU-T X.500 Series of Recommendations, [2-5].

The hardest decision has been made, and a Directory system has been acquired by the organization. Now what should be done? Therein lies the second most difficult task related to a Directory system - designing the schema. What information will be kept in the Directory? Is it primarily for personnel data i.e., name, mailing address, telephone? Is the Directory intended to be used internally i.e., only available to the different branches of the organization? Will the Directory be distributed or centralized? Will it contain property information, device information, customer information, or product information? The answers to these and similar questions are necessary in designing a Directory schema.

This document, specifically Chapter 2, will explain the process of designing a Directory schema. A review of the basic schema components is provided. The details of the schema design process are then discussed. Finally, Chapter 3 reviews several pilot projects to show by example what can be done in designing a Directory schema.

2. Schema Design

This chapter concentrates on Directory schema design. Section 2.1 describes the Directory schema components. The description emphasis is on why the elements are necessary and how they can be used. Section 2.2 provides information on storage of schema information in the Directory Information Tree (DIT). Finally, Section 2.3 describes some of the schema components defined by the Directory standards [2-5] and the process of tailoring these components to meet the needs of a particular organization.

2.1 Directory Schema Components

The Directory Schema constitutes the framework within which Directory information is stored. It consists of a set of rules and definitions which define the naming of entries, the content of attributes and entries, the structure of the Directory as a whole, and the hierarchical relationships between entries. The reasons for a Directory schema include:

- ♦ providing a description of the structure of the data in the Directory,
- ♦ describing the relationships between classes of Directory objects, and
- ♦ providing efficient access to and use of the Directory.

The schema enables the Directory system to prevent such things as:

- ♦ creating subordinate entries which are of the wrong class (e.g., a country name as a subordinate of an organization),
- ♦ adding an entry containing attribute types inconsistent with the entries object class (e.g., a generational qualifier added to a device's entry), and
- ♦ adding an attribute value with a syntax not matching the syntax defined for the attribute-type (e.g., a bit string to a printable string).

The Schema comprises the following components (in accordance with ITU-T Recommendation X.501 Clause 12.2 [2]):

- ♦ Name Form definitions which describe how Directory entries should be named.

  The definition of a name form is accomplished by 1) specifying the named object class, 2) identifying mandatory attributes to be used for Relative Distinguished Names (RDNs), 3) indicating the optional attributes, if any, that may be used for RDNs and 4) assigning an object identifier for the name form.

- ♦ Attribute Type definitions which identify the object identifiers by which the attribute is known.

  Defining an attribute type includes identifying the syntax and associated matching rules, specifying whether the attribute is an operational attribute and, if so, what the type is, specifying whether the attribute is a collective attribute, designating single or multiple values for the attribute and indicating whether the attribute is derived from another attribute type. The assignment of an object identifier for the attribute type is part of the definition of an attribute type.

- ♦ Object Class definitions which delineate the mandatory and optional attributes which are members of a given object class and indicate the kind of object class being defined.

  The assignment of an object identifier for the object class is part of object class definitions.

- ♦ Directory Information Tree Content Rule definitions which allow for the inclusion of attributes into entries not indicated in the entries' structural object classes.

  Every entry in the DIT is governed by at most one DIT content rule, which can be identified by looking at the value of the entries **structuralObjectClass** attribute. If no DIT content rule is defined for a structural object class, then the entries of that class shall contain only the attributes specified by the structural object class definition.

- ♦ Structure rules which define the allowable form of the Directory Information Tree (DIT) hierarchy in terms of object classes and specify the mandatory naming attributes for each object class.

  Each object and alias entry is regulated by a single DIT structure rule. A DIT structure rule definition consists of a 1) unique integer identifier which is unique within the scope of the subschema, 2) the specification of the name form for entries regulated by the DIT structure rule, and 3) the set of the permissible superior structure rules, if necessary.

  Please note that one of the mandatory naming attributes must be part of the relative distinguished name (RDN), but may be augmented with other attribute types. According to the X.500 Directory standard [2]. An entry will frequently contain a single distinguished value. This means that the RDN will be composed of a single type and value pair. However under some circumstances (in order to differentiate), additional values may be used. It is the responsibility of the relevant naming authority to ensure that RDNs of all the entries with a particular immediate superior are distinct by appropriately assigning distinguished attribute values.

- ♦ Matching Rule definitions which define the matching rules to be applied to attributes.

  This includes defining the syntax of an assertion or claim of the matching rule, specifying the different types of matches that the rule supports, defining the appropriate rules for evaluating a given assertion with respect to the target attribute values held in the Directory Information Base. Basic matching rules include equality , ordering, and substring matching.

Designing a Directory schema is an important part of implementing a Directory Service, but it is not a one-time task. The schema document is a "living document" that needs procedures for

- ♦ allowing submission of requests for new attributes and object classes to be added into the schema;
- ♦ allowing groups of object classes and attribute types defined elsewhere to be integrated into the schema; and
- ♦ checking for the redundancy of any previously defined attribute types and object classes.

One method which aids in the administration of the Directory is storage of the schema information in the Directory Information Tree. Storing the local schema information in subentries in the DIT enables schema information to be retrieved by remote DUAs and DSAs. Handling local schema information in this manner allows DUAs and DSAs to properly interpret and manipulate information from a given part of the Directory by reading the schema first.

The storage of local schema information is accomplished through the use of subentries; a subentry is a special kind of entry which is immediately subordinate to an administrative point. A subentry contains attributes that relate to a subtree or subtree refinement associated with its administrative point. A subentry does not have subordinates and it may also contain operational attributes using the appropriate semantics. Subentries must have the attributes of **commonName, subtreeSpecification,** and **objectClass**. The **objectClass** attribute indicates the purpose of the subentry in Directory operation.

2.2 Selecting and Tailoring Schema Components

In defining a schema for an organization, it may be appropriate to use some of the schema components already defined within the Directory standard. The ISO standard, ISO 9594 part 7 [5] contains selected schema components which can be used by organizations as part of their directory schema. This section lists the selected schema components defined in the Directory standard and provides some practical examples of their use.

2.2.1 Name Forms

A name form specifies which attribute types within an object class may form part of the Relative Distinguished Name (RDN) of entries belonging to that class. The Directory standard defines a name form for each object class mentioned below with the exception of **strongAuthenticationUser** and **certificationAuthority**. Each name form definition specifies how entries of a given object class may be named. For example, here is the name form definition for the *country* object class:

```
countryNameForm     NAME-FORM ::= {
        NAMES               country
        WITH ATTRIBUTES     {countryName}
        ID                  {id-nf-countryNameForm}}
```

Most of the name forms for people or object related object classes such as **person, organizationalPerson, organizationalRole, groupOfNames, residentialPerson, applicationProcess, applicationEntitiy, dSA,** and **device** specify the attribute *Common Name.* See ITU-T Recommendation X.521 [5] for the name form definitions provided by the Directory standard.

2.2.2 Object Classes

The object classes provided by the Directory standard are listed below. Each of these object class definitions can be found in the Directory Standard - ITU-T Recommendation X.521 [5]. The object class definition for the **country** object class is included below to show what is contained in an object class definition.

country - the *country* object class is used to define country entries in the Directory Information Tree. The standard definition looks like this

```
country         OBJECT-CLASS    ::=    {
        SUBCLASS OF             top
        MUST CONTAIN            {countryName}
        MAY CONTAIN             {description |searchGuide}
        ID                      {id-oc-country}}
```

locality - the *locality* object class is used to define locality in the Directory Information Tree. The *locality* object class must contain at least one of (Locality Name or State or Province Name).

organization - this object class is used to define organization entries in the Directory Information Tree.

organizationalUnit - this object class is used to define entries which represent subdivisions or organizations.

person - the *person* object class is used to define entries representing people in the generic sense.

organizationalPerson - this object class is used to define entries representing people who are associated with an organization, i.e., employed by the organization or important to the organization.

organizationalRole - the *organizational role* object class is used to define entries which represent an organizational role, i.e., a position or role within an organization.

groupOfNames - this object class is used to define a unordered set of names. This set of names may represent individual objects or other groups of names. The membership of a group is static; it is explicitly modified by administrative action.

GroupOfUniqueNames - the *group of unique names* object class, like the *group of names* object class, defines an unordered set of names. The only difference between these two object classes is that the *group of unique names* object class represents a set of names whose integrity can be assured.

residentialPerson - this object class is used to define entries which represent individuals in a residential environment.

applicationProcess - the *application process* object class is used to define entries representing application processes. An application process is defined as an element within a real open system which performs information processing for a given application.

applicationEntity - this object class is used to define entries which represent application entities. An application entity consists of those items of an application process pertinent to OSI.

dSA - the *DSA* object class is used to define entries representing Directory Service Agents. A DSA is an OSI application-process which is part of the Directory. The DSA provides access to the Directory Information Base (DIB) to Directory User Agents (DUAs) and to other DSAs.
device - this object class is used to define entries representing devices, i.e. any physical unit which can communicate, e.g., a modem, tape unit, disk drive, etc.

strongAuthenticationUser - the *strong authentication user* object class is used in defining entries for objects which participate in strong authentication. Strong authentication, defined in the ITU-T Recommendation X.509 [3], is authentication by means of cryptographically derived credentials and is part of the security definitions for the Directory.

certificationAuthority - the *certification authority* object class is used in defining entries which perform the functions of certification authorities, as defined in ITU-T Recommendation X.509 [3]. A certification authority is a third party trusted by one or more users to create and assign certificates. Optionally, the certification authority may create the users' keys. A certificate is the public key of a user, together with some other information, which is rendered unforgeable by encipherment with the secret key held by the certification authority which issued it. Certification authorities are also part of the procedures for security as it relates to the Directory and applications which use the Directory, e.g., electronic mail applications based on the ITU-T X.400 Series of Recommendations.

Most organizations use some, if not all, of these object classes when designing their Directory schema. The **applicationProcess** and **applicationEntity** object classes are extremely useful to maintain information needed to establish remote connections. The **strongAuthenticationUser** and **certificationAuthority** object classes are needed if the organization plans to provide for digital signatures and other security related procedures.

2.2.3 Attribute Types

Part 6 of the Directory standard [4] defines a set of standard attributes which must be supported by all X.500 implementations and which can be used to develop a Directory schema. These attribute types are organized into broad categories. Listed below are the categories and the attributes contained in each:

<u>Labeling Attributes</u> - **name**, **commonName**, **surname**, **givenName**, **initials**, **generationQualifier**, **uniqueIdentifier**, **dnQualifier**, and **serialNumber**;

<u>Geographical Attributes</u> - **countryName**, **localityName**, **collectiveLocalityName**, **stateOrProvinceName**, **collectiveStateOrProvinceName**, **streetAddress**,

collectiveStreetAddress, and **houseIdentifier**;

Organizational Attributes - **organizationName**, **collectiveOrganizationName**, **organizationalUnitName**, **collectiveOrganizationalUnitName**, and **title**;

Explanatory Attributes - **description**, **searchGuide**, **enhancedSearchGuide**, and **businessCategory**;

Postal Addressing Attributes - **postalAddress**, **collectivePostalAddress**, **postalCode**, **collectivePostalCode**, **postOfficeBox**, **collectivePostOfficeBox**, **physicalDeliveryOfficeName**, and **collectivePhysicalDeliveryOfficeName**;

Telecommunications Addressing Attributes - **telephoneNumber**, **collectiveTelephoneNumber**, **telexNumber**, **collectiveTelexNumber**, **teletexTerminalIdentifier**, **collectiveTeletexTerminalIdentifier**, **facsimilieTelephoneNumber**, **collectiveFacsimilieTelephoneNumber**, **X.121Address**, **internationalISDNNumber**, **collectiveInternationalISDNNumber**, **registeredAddress**, and **destinationIndicator**;

OSI Application Attributes - **presentationAddress**, **supportedApplicationContext**, and **ProtocolInformation**;

Relational Attributes - **distinguishedName**, **member**, **uniqueMember**, **owner**, **roleOccupant**, and **seeAlso**;

Preferences Attributes - The Directory Standard defines another category of attribute types known as the preferences attribute types. These attribute types are concerned with the preferences of an object. The only preference attribute type defined at this time is the **preferredDeliveryMethod**. This attribute type specifies the object's priority order regarding the method to be used for communicating with it. The possible delivery methods which can be specified by this attribute are: any method, mhs, physical, telex, teletex, g3 facsimile, g4 facsimile, ia5 terminal, videotex, and telephone delivery.

2.2.4 Directory Information Tree (DIT) Content Rules

A DIT Content Rule specifies the allowable content of entries of a given structural object class through the specification of an optional set of auxiliary object classes and mandatory, optional and precluded attributes. If an entry permits collective attributes, these must be included in the DIT content rules. The abstract syntax of a DIT content rule is expressed by the following ASN.1 type:

```
DITConentRule          ::=     SEQUENCE {
        structuralObjectClass          OBJECT-CLASS.&id,
        auxiliaries                    SET OF OBJECT-CLASS.&id OPTIONAL,
        mandatory              [1]     SET OF ATTRIBUTE.&id OPTIONAL,
        optional               [2]     SET OF ATTRIBUTE.&id OPTIONAL,
        precluded              [3]     SET OF ATTRIBUTE.&id OPTIONAL }
```

The structural object class to which the DIT content rule applies is specified in **structuralObjectClass**. The **auxiliaries** component represents the auxiliary object class allowed for an entry. The **mandatory** component species user attribute types which an entry shall contain in addition to those which it shall contain according to **structuralObjectClass** and **auxiliary**. The **optional** component indicates user attribute types which an entry may contain in addition to those which it may contain according to **structuralObjectClass** and **auxiliary**. Finally, the **precluded** component specifies a subset of the optional user attribute types of **structuralObjectClass** and **auxiliary** which are precluded from an entry. Each of the above components apply for entries to which the DIT content rule applies.

2.2.5 Matching Rule Definitions

Matching rules are used to evaluate attribute value assertions. The syntax used in the attribute value assertion (i.e., the **assertion** component of the attribute value assertion) is the matching rule's assertion syntax. Matching rules may apply to different types of attributes with different attribute syntaxes. It is important that a matching rule definition contain both the specification of the syntax of an assertion of the matching rule and the way in which values of this syntax are used to perform a match. A matching rule definition which is to be used with attributes with different ASN.1 syntaxes must specify how matches are to be performed. Matching rules are defined as values of the **MATCHING-RULE** information object class:

```
MATCHING-RULE ::= CLASS {
        &AssertionType          OPTIONAL
        &id                     OBJECT IDENTIFIER UNIQUE }
WITH SYNTAX     {
        [ SYNTAX                &AssertionType ]
        ID                      &id }
```

For matching rules defined using the above information object class:

a) **&AssertionType** is the syntax for an assertion using this matching rule; if it is absent, the assertion syntax is the same syntax as that of the attribute the rule is applied to;

b) **&id** is the object identifier assigned to it.

The definition for the **objectIdentifierMatch** matching rule is as follows:

```
objectIdentifierMatch     MATCHING-RULE ::= {
        SYNTAX            OBJECT IDENTIFIER
        ID                id-mr-objectIdentifierMatch }
```

The **objectIdentifierMatch** matching rule is an equality matching rule. This means that a presented value of type object identifier must match a target value of type object identifier in the number of integral components and each integral component of the first is equal to the corresponding component of the second.

The basic matching rule definition types provided are equality and partial matching rules. Within this break-down there are then several categories of matching rules - string, syntax-based, time, first component and word matching rules. The string matching rules, for example, include such definitions as:

1) Case ignore
2) Case ignore ordering
3) Case ignore substring
4) Case exact match
5) Case exact ordering
6) Case exact substrings
7) Numeric string
8) Numeric string ordering
9) Numeric string substring
10) Case ignore list
11) Case ignore list substrings

In string matching rules spaces are ignored; this includes leading spaces, trailing spaces and multiple consecutive internal spaces. Therefore, strings to be matched are matched as if the insignificant spaces were not present in the string.

For more information on matching rule definitions, particularly definitions provided by the Directory standard, please refer to the Directory standard, part 6 [4].

2.2.6 Directory Information Tree Structure Rules

A Directory Information Tree (DIT) structure rule is a mechanism provided by the subschema administrative authority which the Directory uses to control the placement and naming of entries within the subschema. It is common for the subschema governing a subtree of the DIT to contain several DIT structure rules permitting several types of entries within the subtree.

The abstract syntax of a DIT structure rule is expressed by the following ASN.1 type:

```
DITStructureRule ::=          SEQUENCE {
        ruleIdentifier                  RuleIdentifier,
                                        -- must be unique within the scope of the subschema
        nameForm                        NAME-FORM.&id,
        superiorStructureRules          SET OF RuleIdentifier OPTIONAL }

RuleIdentifier        ::=     INTEGER
```

The **ruleIdentifier** component of the DIT structure rule identifies the DIT structure rule uniquely within a subschema. The **nameForm** component specifies the name form for entries governed by the DIT structure rule. The **superiorStructureRules** element identifies permitted superior structure rules for entries governed by the rule. It the **superiorStructureRules** component is absent, the DIT structure rule applies to an autonomous administrative point.

The Directory standard provides a definition for the **STRUCTURE-RULE** information object class to facilitate the documentation of DIT structure rules. The definition of the **STRUCTURE-RULE** information object class is as follows:

```
STRUCTURE-RULE        ::=     CLASS {
        &nameForm                       NAME-FORM
        &SuperiorStructureRules         STRUCTURE-RULE  OPTIONAL,
        &id                             RuleIdentifier UNIQUE }
WITH SYNTAX {
        [ NAME FORM                     &nameForm ]
        [ SUPERIOR RULES                &SuperiorStructureRules ]
        ID                              &ruleIdentifier }
```

2.3 Local Schema Components

From the previous sections, it can be seen that the Directory standard provides a wide variety of object classes, name forms and attribute types which may be used in defining a schema. Directory schema elements can be augmented by various implementers bodies, by vendors, or by users. While it is essential that organizations be able to extend the Directory schema to meet their specific needs, problems with interoperability and duplication of effort are likely to arise if such extensions are carried out independently. Some coordination of effort is necessary.

This section addresses the subject of tailoring Directory schema elements to meet the organization's needs and consists of several examples which should provide insight into how an organization might determine that it needs to augment the schema elements provided by the Directory.

When tailoring schema elements, new elements like object classes and attribute types are derived from the standard provided definitions or more generic definitions. An object class (subclass), for example, may be derived from an object class (direct superclass) which itself may be derived from an even more generic object class. In such a case, the subclass "inherits" the mandatory and optional attribute specifications of its superclass. An object class derived from 2 or more direct superclasses displays the principle of multiple inheritance, because the derived object class "inherits" the mandatory and optional attribute specifications of all of the direct superclasses.

New attribute types may be derived in the same manner, i.e., when defining a new attribute type, it is possible and often desirable to derive the attribute using the characteristics of some more generic attribute type. The new attribute type is then a "direct subtype" of the more generic attribute type or the "supertype" from which it is derived.

Tailoring of schema elements should be done anytime the organization chooses to place organization-specific information in the directory. For example, the "Cartoons Are US" company is installing a Directory service. The directory will maintain all the usual information such as name, address, telephone number, etc. for most of the employees. But, the company decides to create an object class **cartoonist** to capture information specific to those employees who are cartoonists. Once again, all the normal information is there (name, address, telephone number, etc.), but a new attribute type has been created as well, **cartoonCharacter.** The attribute type, **cartoonCharacter**, holds the name of the cartoon character for which the cartoonist is responsible. For example, cartoonist Don Jones is responsible for the cartoon character, "Tiffany Termite".

Tailoring is necessary if the organization chooses to place information into the directory which originally was not thought of as applicable to a Directory service. For example, the PSI White Pages project, mentioned in Chapter 3, decided to store information about publications. Two object classes were created, **documentSeries** and **document**. The object class, **documentSeries**, was used to describe a series of documents. The information stored was the name, description, a reference to another place, the telephone number and other contact information related to the location of the document series. The object class, **document**, was used to describe a specific document. Information stored included the document number, location, document title, version number, author's name, key words, abstract, subject and information about whether this document replaced a previous document or was an update to an existing document.

A pilot X.500 project, the X.500 Integration Pilot, organized by the Corporation of Open Systems (COS) recently published information related to the pilot [10-13]. The pilot was based on implementing an X.500 Directory service for the Southern Company. The Southern Company is the parent firm of one the nation's largest electric utility groups and includes five utility companies. In addition, there are several subsidiaries and facilities involved. The X.500 Directory Service was chosen as a means to exchange information between the proprietary mail and directories used by the Southern Company. The use of X.500 also allowed the Southern Company to share additional information with other utilities and business partners by allowing access to each other's corporate directories.

The pilot utilized all of the object classes and attributes defined by the Directory standard. Several additional object classes were defined. One of the object classes defined, **southernCompanyPerson**, was derived from the Directory standard object classes of **person** and **organizationalPerson**. The Electric Power Research Institute (EPRI) also used these object classes to define a company-specific object class, **eprinetPerson**. This object class contained the following attributes:

distinguishedName
commonName
surname
title
description
telephoneNumber
facsimilieTelephoneNumber
postalAddress
mhs-or-address

The object class **groupOfNames** was used by pilot participants to store information about the Disaster Recovery Team at each of the participating companies. The **groupOfNames** object class contained the attribute types of **commonName**, **member**, **description**, and **seeAlso**. The EPRI also created an attribute type, **jpegPhoto**, to store peoples pictures.

Tailoring schema components is a common practice and quite easy to do. As noted above, the definitions for most of the tailored object classes were modeled after the standard. This is also true of two of the pilots reviewed later in Chapter 3. In one of the pilots, The National Institute of Standards and Technology (NIST)/General Services Administration (GSA) X.500 pilot, object classes were customized to be more appropriate for a government organization. For example, the object class **govOrganization** is similar to the standard defined **organization** object class, but **govOrganization** adds three attributes to its definition, **mhs-or-addresses, rfc822Mailbox,** and **proprietaryMailbox**. The additions were made because of the increasing use and dependence on electronic mail. The **proprietaryMailbox** attribute type is a new attribute type created using the definitions of **mhs-or-addresses** and **rfc822Mailbox**, but it is used to hold the address for a proprietary electronic mail system.

3. Directory Pilots and Projects

This section describes a selection of X.500 pilot projects and other Directory related efforts which may be helpful to organizations interested in implementing the Directory and developing a Directory schema. Some of the projects are ongoing, while others have been completed and the status of others is unknown. Whatever the status of the pilot or project, each provides valuable insights into the development of Directory schemas.

3.1 The Electronic Mail Program Management Office (PMO) Directory Guidelines

In July 1993, the Office of Management and Budget (OMB) chartered an Electronic Mail Task Force (EMTF) to examine the state of electronic messaging among Federal agencies and to make recommendations on what should be done to provide interoperable business-quality electronic mail throughout all Federal government agencies.

In April 1994, the EMTF issued its final report defined business-quality E-mail as "*a service that appears to the user to be a single, unified electronic postal system that offers robust and trustworthy capabilities with legally-sufficient controls for moving all forms of electronic information among employees at all levels of government, and with the public we serve; and, like the nation's telephone network, is affordable, ubiquitous, efficient, accessible, easy-to-use, reliable, cost-effective and supported by an effective directory service.*" The recommendations of the EMTF included the following:

1. require Government-wide E-mail connectivity,
2. establish a Government-wide E-mail Directory,
3. establish an E-mail Program Office.

The EMTF report recommended that the Department of Defense (DoD) Defense Messaging System (DMS) operational characteristics specifications be an important source of information to be used when defining the operational characteristics of business-quality Government-wide E-mail. The EMTF recognized however that Federal Agencies outside of the DoD may have different requirements for business-class messaging than the Department of Defense. The EMTF recommended that an Electronic Mail Program Office (PMO) be established to coordinate, shape and implement policies that would provide electronic interoperability among Federal agencies and between each Federal agency and the outside community that it serves. The OMB chartered and funded a group within the General Services Administration (GSA) to perform this task.

The GSA E-mail PMO has developed a two-year plan to provide business quality messaging to Federal agencies by 1997. Government-wide business quality messaging however will not happen without the active participation of experts from all Federal agencies. The GSA E-mail PMO will act as coordinator and catalyst to bring Federal agency experts together to develop the detailed functional documents that specify Federal agency electronic mail and directory service requirements. The GSA E-mail PMO will also be responsible for removing all barriers that would inhibit the implementation of that strategy.

Directory services, as currently envisioned by the GSA E-mail PMO, will be provided by implementations based on the ITU-T X.500 Series of Recommendations. It is in the interest of each Federal agency to align their requirements for business quality messaging with those of other agencies. The E-mail PMO has established the means by which the necessary inter-agency interaction can occur. The E-mail PMO has set up two on-line forums to facilitate communication among Federal agencies for the purpose of achieving a consensus on required E-mail services. Federal agency personnel can subscribe to one or both on-line forums by sending an e-mail message to listproc@etc.fed.gov with

```
SUBSCRIBE email-1 NAME OF SUBSCRIBER
and/or
SUBSCRIBE X.500-1 NAME OF SUBSCRIBER
```

in the body of the message. To receive a list of additional on-line groups that deal with related issues, put LIST in the body of the message. In addition, a volunteer working group called the Electronic Messaging and Directory Working Group (EMADWG) has been formed to address a large range of messaging issues at the working level. For further information about E-mail PMO sponsored activities, contact Jack Finley at Jack.Finley@gsa.gov.

Each Federal agency should work with other Federal agencies to develop a directory service schema to support business quality messaging. The E-mail PMO is working with independent contractors to develop a directory services schema that can be used by all Federal agencies. The Directory Services schema that is being developed by the E-mail PMO will be available in draft form for comment and approval by Federal agencies. Federal agencies should ensure that all requirements that they have in common with other agencies are incorporated into the final version. The previously mentioned X.500 on-line forum and the EMADWG will play an important role in the review of the draft document. Agencies should understand that the schema that is being developed will allow them to add object classes and attribute types of particular interest to their agency.

Each Federal agency must determine how the directory service will be administered within the agency. The DIT can be broken up into subtrees. The E-mail PMO will administer the portion of the DIT dealing with the U.S. Government. The E-mail PMO, under its authority to administer the portion of the DIT under **country** = US, **organization** = U.S. Government, will assign **organizational unit** names to requesting government organizations and implement a knowledge reference DSA which will contain the address of all DSAs that implement the Directory Service at the organizational unit level of the Directory Information Tree.

Each organization that requests an organizational unit name from the E-mail PMO must decide how to structure and administer the subtree in the DIT that contains the assigned **organizational unit** as its root entry. The requesting organization must know the extent of its administration authority. For example, does the Department of Treasury have the authority to administer the name space for the Internal Revenue Service? If so, then the Internal Revenue Service becomes a subordinate **organizational unit** under the Department of Treasury. Does the Internal Revenue Service have the authority to request and administer its own **organizational unit** subtree? There is no correct answer but policy must be determined before a request for the assignment of an **organizational unit** name is made.

Once the extent of the administration authority of an organization is determined, the next issue to be determined is how the administrative authority is to be partitioned within the organization. Each agency must determine how many DSAs will provide users with transparent access to the data stored in their portion of the DIT and what part of the DIT, or what naming context, is the responsibility of each DSA.

3.2 National Institute of Standards and Technology (NIST)/General Services Administration (GSA) Pilot Project

In July 1991, the National Institute of Standards and Technology (NIST) issued a report entitled *DRAFT NIST/GSA X.500 Pilot Project: Schema Definition* [6]. The report detailed the schema to be used for the NIST/GSA pilot directory project. This section will describe some of the schema components selected or developed for the NIST/GSA pilot. The schema was composed of four parts: object class definitions, structure rule definitions, attribute type definitions and attribute syntax definitions. (Note: the NIST/GSA pilot was based on the 1988 Directory Standard.)

This particular project illustrated the concept of selecting and tailoring schema definitions to meet the needs of an organization. The NIST/GSA Pilot schema includes all the object classes defined in the Directory standard. In addition, several were derived from the standard object classes. A description of the derived object classes follows:

govOrganization - this object class adds MHS and RFC822 electronic mail addresses to the **organization** object class, as well as an attribute for a proprietary mail system, **proprietaryMailbox**.

govOrganizationalUnit - this object class adds MHS and RFC822 electronic mail addresses to the **organizationalUnit** object class, as well as an attribute for a proprietary mail system, **proprietaryMailbox**.

govOrganizationalPerson - this object class adds MHS and RFC822 electronic mail addresses to the **organizationalPerson** object class, as well as an attribute for a proprietary mail system, **proprietaryMailbox**.

govOrganizationalRole - this object class adds MHS and RFC822 electronic mail addresses to the **organizationalRole** object class, as well as an attribute for a proprietary mail system, **proprietaryMailbox**.

service - the **service** object class is used to define entries representing offered Open System Interconnection (OSI) data communications service. The entry contains information about where a service is offered (i.e., the presentation address and the computer) and details about the service capabilities.

computer - the **computer** object class is used to define entries representing real computers. This object class contains the name assigned to the computer, the identity of the contact point and may optionally contain information such as its operating system, serial number, and supported services.

modem - the **modem** object class is used to define entries representing modems. This object class contains a contact point for the modem and contains information about the host computer, the telephone number and the physical location.

printer - the **printer** object class is used to define entries representing printers. This object class once again has a contact point designated and contains information on the hosting computer, the printer speed, supported fonts, supported character sets, physical location, and the page description language.

The next paragraph describes the structure rules and naming guidelines used by the NIST/GSA pilot project. It is important to understand that the NIST/GSA pilot was based on the 1988 Directory standard. The current Directory standard, 1994, has different procedures for structure rules, etc.

The 1988 Directory standard suggests a set of structure rules and naming guidelines. The NIST/GSA pilot adopted these suggestions. Therefore, most of the subordinate data for the object classes can be found in the 1988 Directory standard. The exceptions are the object classes created to represent organization specific needs. This is true both for object classes considered as subordinates of other object classes and in the specification of their subordinates. Below is the subordinate information for the tailored object classes:

- **country** has **govOrganization** as a subordinate
- the subordinates of **govOrganization** are **govOrganizationalUnit**, **govOrganizationalPerson**, **govOrganizationalRole**, **locality**, **groupOfNames**, **applicationProcess**, **computer**, **device**, **dSA**, **modem**, **printer**, and **service**
- **locality** has **govOrganization**, **govOrganizationalUnit**, **govOrganizationalPerson**, **govOrganizationalRole** as subordinates
- the subordinates of **govOrganizationalUnit** are **govOrganizationalUnit**, **govOrganizationalPerson**, **govOrganizationalRole**, **locality**, **groupOfNames**, **applicationProcess**, **dSA**, **computer**, **device**, **modem**, **printer**, and **service**
- the subordinates of **govOrganizationalPerson** are **computer**, **device**, **modem**, **printer**, and **service**
- the subordinates of **govOrganizationalRole** are **locality** **govOrganizationalUnit**, **computer**, **device**, **modem**, **printer**, **service**, and **dSA**
- the subordinates of **computer** are **device**, **modem**, **printer**, and **service**
- the subordinates of **modem** are **common name**
- the subordinates of **printer** are **common name**
- the subordinates of **service** are **common name**

The third schema component of the NIST/GSA pilot consists of attribute type definitions. The NIST/GSA X.500 pilot schema includes all the attributes defined in the Directory standard. Additionally, some attributes were defined in order to support the pilot schema object classes defined above. Below is the list of attribute types created for the pilot:

supportedProfiles - this attribute specifies the profiles supported by the service, e.g., **supportedProfiles** = TELNET, TRANSPORT.

supportedAbstractSyntaxes - this attribute specifies the abstract syntaxes supported by the service, e.g., **supportedAbstractSyntaxes** = FTAM-FADU, FTAM-PCI

hostingComputer - this attribute denotes the distinguished name of the computer system that is hosting the service. This attribute may not be needed if an entry is immediately subordinate to an entry of object class computer; in this case, the relationship is derived from the relative positions of the entries in the tree. For example, **hostingComputer** = /C="US"/O="GOV"/OU="Commerce" /commonName="Admin7"/

operatingSystem - this attribute identifies the operating system running on a particular computer, e.g., **operatingSystem** = "4.4 BSD UNIX"

supportedServices - this attribute denotes the set of services available on the computer. Note that the set of services available on a computer system can be identified two ways. First, reference can be made to them using this attribute type. Alternatively, the entry for the service can be placed directly under the entry for the computer in the DIB, in which case this attribute would not need to be used. For **example, supportedServices** = /C="US"/O="GOV"/OU="Commerce"/OU="NIST" /commonName="ftam-responder"/

contactPoint - this attribute identifies the entries of those responsible for equipment, e.g., computers, printers, and modems. This could be; for example, the owner, a system administrator, or a property officer. For example, **contactPoint** = /C="US"/O="GOV"/OU="Commerce"/OU="NIST" /CN="Smith"/

printerSpeed - this attribute identifies the speeds at which a printer operates, e.g., **printerSpeed** = "600 lpm"

supportedFonts - this attribute identifies the fonts supported by a printer, e.g., **supportedFonts** = "computer modern", "american modern", "CM", "AM"

supportedCharacterSets - this attribute identifies the character sets supported by a printer, e.g., **supportedCharacterSets** = "ASCII"

physicalLocation - this attribute specifies the location of an object. It is the name of a place which is commonly known in some limited scope, e.g., **physicalLocation** = "Building 225", "225", "Technology/B217"

pageDescriptionLanguage - this attribute specifies the page description language supported by the object (printer) whose entry contains this attribute, e.g., **pageDescriptionLanguage** = "PostScript", "OUIC", "CAT"

proprietaryMailbox - this attribute is used to store electronic mail addresses other than those that would be put into the **rfc822Mailbox** and **mhs-or-addresses** attributes. These mailboxes will be on specific networks operating proprietary mail protocols. Thus, in addition to the mailbox name, the network name and protocol are also needed. Values of the attribute are strings having, by convention, an internal structure comprising '$'-separated fields. For example, the structure would be <network name> '$' <mail protocol> '$' <mailbox name> and possible values might be **proprietaryMailbox** = "NISTnet $ IBM Profs $ rcolella"

The NIST/GSA X.500 pilot schema also imported attribute types from other sources. These included the **mhs-or-addresses** and **rfc822Mailbox** attribute types. The **mhs-or-addresses** attribute type is imported from the Message Handling Service standard (MHS). This attribute specifies the MHS Originator/Recipient (O/R) address to use when sending mail to the object denoted by the entry which contains this attribute. The **rfc822Mailbox** is the electronic mail address in the form used by the Internet. This attribute is provided for support of the Simple Mail Transfer Protocol (SMTP) in the Internet community by X.500.

The NIST/GSA X.500 pilot schema includes all of the attribute syntaxes defined in the 1988 Directory standard. No new attribute syntaxes were defined for the pilot, but a two were imported, namely the syntaxes for **mhs-or-addresses** and **rfc822Mailbox.**

3.3 The PSI White Pages Pilot Schema

In July 1989, NYSERNet, Inc., a non-profit company running a regional network on the Internet, started a Directory pilot [7-9]. Since that time the pilot has been continued by Performance Systems International (PSI). The PSI has been extending and evaluating the service in response to experience gained during the operation of the PSI White Pages project. The goals of the pilot were

- to provide a large distributed information service involving administration by multiple organizations,
- to conduct the first production-quality field test of the Open Systems Interconnection (OSI) Directory, and
- to establish the first large-scale production application of OSI technology on top of the Internet suite of protocol.

The pilot used THe Obviously Required Nameservice (THORN) architecture as the basis for its schema. The THORN architecture defines additional object classes and attribute types to be used for the pilot. Listed here are some of the new object classes and attributes types recommended by THORN - **friendly_CountryName, homePhone, homePostalAddress, manager, masterDSA, slaveDSA, mobileTelephoneNumber, pagerTelephoneNumber** and **secretary**.

The object class, **friendlyCountry**, for example, provides a user-friendly identification of sovereign nations and this object class is subordinate to the **country** class. Possible values for the attribute **friendlyCountryName** which is part of the **friendlyCountry** object class are for **country** = CH, "Switzerland" or "CH".

Below are some of the new attribute types defined for the pilot:

favoriteDrink - a string describing the user's favorite drink, e.g., **favoriteDrink** = "Iced Tea"
homePhone - a string giving the user's phone number using the international notation, e.g., "+1 401-555-1212"
homePostalAddress - a string providing the mailing address of the person's home
info - a string containing additional, textual information about the person, e.g., "Trekker, Good Cook"
mobileTelephoneNumber - a string describing the user's mobile telephone number (e.g., for a cellular phone)
otherMailbox - a string containing the user's computer mail address in various domains
photo - facsimile bitmap of the user's face
rfc822Mailbox - a string containing the user's computer mail address in the Internet
secretary - a string which provides the Distinguished Name of the user's administrative support
userClass - a string describing the user's classification, e.g., "staff"
userid - a string containing the user's login name, e.g., "jsmith"

Several of the object classes defined in the architecture and for the pilot are interesting and may provide insight into the process of schema design. A very fundamental object class defined is **thornObject**. This object class is used as a superclass to define several other object classes to model THORN objects. One of the most relevant object classes defined is **thornPerson** which is a subclass of **thornObject** and contains the attributes **userid, textEncodedORAddress, rfc822Mailbox, favoriteDrink, roomNumber, userClass, homePhone, homePostalAddress, secretary**, and **personalTitle.** The object class, **thornPerson**, is also a subclass of **person** and therefore displays the property of multiple inheritance.

Another object class defined for the Directory pilot is **pilotPerson**. The **pilotPerson** object class was created to aid the administrators of the pilot who wanted a few more attributes to capture information about persons in the Internet. The **pilotPerson** object class is a subclass of **thornPerson**. This object class may contain the **localAttributeSet, postalAttributeSet, telecommunicationsAttributeSet, businessCategory, title, otherMailbox, mobileTelephoneNumber,** and **pagerTelephoneNumber**.

From these three object classes it is possible to see a progression:

- ♦ from **thornObject** which captures information about a class of objects
- ♦ to **thornPerson** which merges that information with generic personal information; and finally,
- ♦ to **pilotPerson** which merges all of that information with additional information specific to people in the pilot.

This shows how the extensibility facilities in the Directory's information framework can be used.

This project is also interested in experimenting with document identification and retrieval using the Directory.

The pilot defined two object classes to help with this aspect: **documentSeries** and **document**. The **documentSeries** object class describes a series of documents under a common administration. The object class is a subclass of TOP and must contain a **commonName**. The attributes which may be included are **description, seeAlso, telephoneNumber, localityName, organizationName,** and **organizationalUnitName**.

The **document** object class is intended to describe an individual document. It is a subclass of **thornObject** and it must contain a **documentIdentifier**. The **document** object class may contain any of the following attributes: **commonName, description, seeAlso, localityName, organizationName, organizationalUnitName, documentTitle, documentVersion, documentAuthor, documentLocation, obsoletesDocument, obsoletesByDocument, updatesDocument, updatesByDocument, keywords, subject, abstract, authorCN, authorSN,** and **documentStore**.

This project is an excellent example of tailoring the X.500 Directory schema components to meet the needs of the organization. More information on this project and the THORN X.500 Naming Architecture can be obtained from the North American Directory Forum (NADF) at +1 202-342-2727 or see References [7-9].

4. Conclusion

This paper presents a high level description of X.500 Directory schema design which should be helpful to organizations preparing to utilize a Directory service. The examples presented here provide information on both standard-defined schema elements and tailored or organization-defined schema elements. This should aid organizations as they attempt to define the schema for their Directory service. For more information, please refer to the references provided. These should provide the additional detail schema designers require.

REFERENCES

1. *Guidelines for the Evaluation of X.500 Directory Products*, National Institute of Standards and Technology, Special Publication 500-228, May 1995.

2. ITU-T Recommendation X.501 (1993) | ISO/IEC 9594-2: 1993, *Information Technology - Open Systems Interconnection - The Directory: Models.*

3. ITU-T Recommendation X.509 (1993) | ISO/IEC 9594-8: 1993, *Information Technology - Open Systems Interconnection - The Directory: Authentication Framework.*

4. ITU-T Recommendation X.520 (1993) | ISO/IEC 9594-6: 1993, *Information Technology - Open Systems Interconnection - The Directory: Selected Attribute Types.*

5. ITU-T Recommendation X.521 (1993) | ISO/IEC 9594-7: 1993, *Information Technology - Open Systems Interconnection - The Directory: Selected Object Classes.*

6. *Draft NIST/GSA X.500 Pilot Project Schema Definition*, The National Institute of Standards and Technology, July 1991.

7. Rose, Marshall T., *The Little Black Book - Mail Bonding with OSI Directory Services.* Prentice Hall, 1992.

8. Kille, Stephen E., *The THORN X.500 Naming Architecture.* THORN Project Internal Report UCL-45, University College London, January 1989.

9. *The THORN and RARE X.500 Naming Architecture.* Report of Computer Science Research Notes 89/48, University College London, May 1989.

10. *Directory Services: A Transition and Co-existence Approach*, X.500 Integration Pilot Project, Phase I Implementation Report, The Corporation for Open Systems, October 1993.

11. *Directory Services: A Transition and Co-existence Approach*, X.500 Integration Pilot Project, Phase II, Stage 1 Implementation Report, The Corporation for Open Systems, June 1994.

12. *Directory Services: A Transition and Co-existence Approach*, X.500 Integration Pilot Project, Phase II, Stage 2 Implementation Report, The Corporation for Open Systems, September 1994.

13. *Directory Services: A Transition and Co-existence Approach*, X.500 Integration Pilot Project, Phase II, Stage 3 Implementation Report, The Corporation for Open Systems, March 1995.